

# LINUX EXPERT

## FAQ

**Q** Using the `ps` command to monitor processes is confusing because of all the command-line switches, and it isn't interactive anyway. Is there a better way to see what's running?

**A** Try the top command. This produces an interactive process listing, sorted by various criteria such as which processes are using the most memory, processor and so on. The listing updates itself every two seconds, which should be adequate for most system monitoring activities.

Press the ? key while top is running to display a list of the hotkeys you can use to control its output format. These include sorting the listing by memory usage (press M) and processor (press C).

Two useful hotkeys enable you to use top to track what a user has running and even kill individual processes. First, you can filter the output so that it shows only the processes owned by a single user. Simply press the U key and enter a username. You can reverse this filtering by pressing U again and hit return.

Second, you can kill a process by pressing the K key and entering the process ID (PID). Top will ask for a signal to send to the process to tell it to stop. The default is signal 15, which tells it to stop immediately.

**Jon Thompson**  
UNIX/Linux management  
and security consultant



linux@computershopper.co.uk

# The open-source movement

What exactly is open-source software, and how is it responsible for the growth of Linux? Jon Thompson explains

**W**hat does open source mean? Is it simply free software written by altruistic individuals and, if so, how have some of those involved in it become millionaires? In this month's *Linux Expert*, we're going to look at what open source software is, how it grew and spawned the Linux phenomenon and how you can get in on the action.

## OPEN SEASON

In September 1983, programmer Richard Stallman announced his intention to quit the famous AI Lab at the Massachusetts Institute of Technology (MIT). He planned to write a complete replacement for the UNIX operating system. He decided to call his offering GNU, a recursive acronym of GNU's Not UNIX, and he intended to give it away. At the time, few understood how he could make a living this way. Twenty years later, the Free Software Foundation he began is thriving and he's still at the helm. His ideas about open-source software form the basis of an industry that threatens Microsoft's domination of the market.

To understand how this has happened, we need to go back to the 1960s. Computing experienced an explosive growth during that decade, due in part to the introduction of a machine called the minicomputer. Reduced from the size of a building to the size of a wardrobe, minicomputers revolutionised computing by making it easier to own your own hardware.

It was usual to bundle the source code for the operating system with a minicomputer, just as it had been with larger mainframes. In places such as MIT the word 'hacker' emerged, describing programmers whose job it was to have the departmental minicomputer accomplish whatever task was required of it. They hacked the operating systems and applications as required, and even abandoned them to write their own. Hackers freely swapped their work with each other and with colleagues at other universities. The predominant atmosphere was one of solving problems and overcoming the limitations of the new hardware.

By the end of the 1970s, however, the freewheeling hacker culture at MIT was under threat. Investors realised there was money in some of the software developed there, as well as the research and products it made possible. Outsiders would have to pay for software that had, up until then, been free.

## FREE REIGN

In his GNU Manifesto, Stallman denounced MIT's new attitude to software as "dishonourable". He thought that while others were free to write for commercial gain, he should be free to do with his software whatever he liked. He would demonstrate this, he said, by programming a free replacement for the UNIX operating system.

At the time, the market for UNIX and the software that ran on it was large. When a company upgraded to a new computer, even from a new supplier, the use of UNIX ensured that it didn't have to rewrite its entire software library. But Stallman, a prolific hacker, knew he

could reproduce much of the functionality of the system libraries, utilities, shells, editors, compilers and debuggers. These would sit on top of a UNIX-compatible kernel.

Stallman had a basic system up and running, along with his own Emacs editor, a C compiler and a few other pieces of the jigsaw, including the free X-Windows graphical user interface also developed at MIT. In January 1984, he quit his job at MIT to devote himself to the GNU project. However, Symbolics Inc, a company to which he had given a self-penned Lisp interpreter, stung Stallman by not allowing him to see the improvements they had made to it, preventing him incorporating them back into the original work.

To stop this happening in future, Stallman came up with the idea of 'copyleft'. The GNU Public Licence (GPL) he wrote, which covers every piece of GNU software, embodies the ideas of copyleft and defines four basic freedoms:

- The freedom to study the software, including its source code
- The freedom to share it with others
- The freedom to modify it
- The freedom to distribute derivative works

If you use a piece of GNU code in one of your own programs, the GNU parts must be available for study by anyone, along with any improvements you've made, so the GNU project can reincorporate them back into the original source code. If you protect your own program using the GPL, you release the source code too, and copyleft ensures that no-one can prevent you obtaining any improvements they've made. Because of this, you'll benefit from their expertise as much as they benefit from yours.

As interest in his idea caught on, Stallman began to publicise GNU and the principles behind making source code available to whoever wanted it. Those needing help with GNU software could also buy technical support, just as they would from a UNIX supplier. Gradually, the project became self-funding. In 1985 Stallman created the non-profit Free Software Foundation to preach the idea of software freedom and to give a management structure to the GNU project. The only problem was his choice of kernel.

## FOLLOWING THE HURD

At the heart of every operating system sits its kernel. This shares slices of processor time, manages virtual



◀ The principles behind 'copyleft' might seem paradoxical, but they've done more to push software forward than proprietary development alone



memory requirements and communicates with peripherals including disks, printers and keyboards. Originally, GNU used a free kernel called Trix, developed at MIT. Although this was broadly compatible with the UNIX kernel, the GNU project eventually abandoned it as being too limiting for its needs.

In its place, it used an adaptation of the equally free and UNIX-compatible Mach kernel, written by Carnegie Mellon University. The GNU project began developing its Mach-based kernel in 1990. Called Hurd, this still forms the basis of the GNU operating system today. But this wasn't the only kernel under development that could run GNU software.

At the University of Helsinki, a student called Linus Torvalds had been working on a kernel of his own as a hobby. He released the first version on to the internet in September 1991. A second version, released a month later, implemented enough functionality that it would run the GNU Bash command-line shell. To encourage others to help develop his kernel, Torvalds decided to release this second version under the GPL. This opened the door for an informal army of programmers to adapt and improve it, with Torvalds benefiting from their work.

Developers the world over realised they were free to use, modify, pass on and even sell open-

source software, as long as they did it under the terms of the licence with which it came. Unfortunately, there were lots of variations of this licence. With Netscape releasing the source code for its Navigator web browser in 1998, there was plenty of scope for confusion. Was Navigator still commercial? Could a developer use part of its source code, or would Netscape sue him for his trouble?

In response to the growing muddle, several open-source advocates held a meeting at Palo Alto, California, in 1998. The delegates, including a man called Eric S Raymond, wanted to create a standard definition for open-source licensing, so developers would understand what rights they had over their own work and the work of others.

The meeting spawned the Open Source Initiative (OSI), which acts as a rallying point and protects the interests of open-source software and its developers. The OSI defines 10 principles to which any licence must conform if it is to be officially open source; for details, see the 'Perfect 10' box below.

### CHAOS THEORY

With so many individuals coding and testing at once, it's sometimes difficult to see how open-source projects don't fall into chaos. In his 1997 book *The Cathedral and the Bazaar*, Raymond

described two methods of running software projects. These he called the cathedral model and the bazaar model. The cathedral model of development is like traditional software development. A dedicated team controls and develops the software and each member has a clearly defined role. In the bazaar model of development, roles are more fluid and anyone can join in. Three major models of open-source project management have emerged from these ideas, offering different levels of control over the 'official' distribution of code.

The first model is BSD-style, named after the style of UNIX development at the University of California at Berkeley. This sees a small, closed core of dedicated developers producing open-source code for distribution. Users can take, use, modify and redistribute this and derivative code, but normally the development team sticks to its own plan and doesn't incorporate modifications made by others.

Second, there is the slightly more open Apache-style of development, used to create the web server of the same name. Like BSD-style, this has its own core development plan, but also incorporates significant modifications to the source code made by private developers.

Significantly, because of the open source nature of these two models, any programmer (or

## Perfect 10 Defining open source

You're free to distribute your programs under whatever terms you like, but there are some special conditions you must meet if you opt to distribute them as 'open source'. The Open Source Initiative (OSI) has published a definition, which consists of 10 points to which your licence must conform.

An increasingly large number of software licences conform to these points. One is Richard Stallman's GNU General Public Licence. This is now in its third revision, and you can read it at [www.gnu.org/copyleft/gpl.html](http://www.gnu.org/copyleft/gpl.html). Another is the Apache licence ([www.apache.org/licenses](http://www.apache.org/licenses)), which protects the web server of the same name. The OSI's points are as follows:

### 1 REDISTRIBUTION

You or anyone else must be free simply to pass your work on to others, whether you charge for it or simply give it away.

### 2 SOURCE CODE

The program you distribute must be available as a source code distribution, as well as in compiled form, so people can inspect the code and build their own binaries. The OSI says the source code must be in "the preferred form in which a programmer would modify the program". This means you can't try to hide what your program does behind densely written code.

### 3 DERIVED WORKS

The licence must allow whoever obtains a copy of your software to make modifications and produce their own programs using it.

### 4 INTEGRITY OF THE AUTHOR'S SOURCE CODE

To prevent confusion between the original

author's software and that modified and released by an independent third party, the licence may demand that third-party releases carry a different version number or other identifier. You can also demand that developers feed back improvements to you for incorporation into the official release. This ensures you have control over the identity and future of your software, while still allowing others to improve it for you.

### 5 DISCRIMINATION

Your licence must not restrict the use of your software to an individual or group, or exclude certain sections of society from using it.

### 6 FIELDS OF ENDEAVOUR

The licence must not prevent anyone using the program in a particular way. For example, you cannot restrict its use in an area of research that goes against your particular belief or preference.

### 7 DISTRIBUTION OF LICENCE

The rights attached to the program by your licence must apply to everyone who uses the software. You cannot decide that people who did not obtain it directly from you can only use or distribute it under a different licence to the one under which you distributed the original.

### 8 LICENCE MUST NOT BE SPECIFIC TO A PRODUCT

The rights attached to your software cannot depend on it being part of a particular product or distribution. You cannot stipulate that if it is included in a major Linux distribution then the licence terms are different to when it forms part of a different product. You can only protect your software with a



▲ Richard Stallman, the man behind the GNU project, and the first to promote freedom for software, arguably kicked off an entire industry by simply giving things away

single licence, so if someone extracts the source code from that distribution and redistributes it, it still carries your original licence terms.

### 9 LICENCE MUST NOT RESTRICT OTHER SOFTWARE

This ensures that you can't insist on people distributing your work only with other open-source software, or that it must not run in conjunction with commercial products.

### 10 LICENCE MUST BE TECHNOLOGY-NEUTRAL

You can't insist that your software must not be ported to a particular piece of hardware, or license it in such a way as to prevent it being used under, for instance, GNOME rather than the KDE desktop. No matter what you might think of Microsoft, you can't restrict your software to run only on Linux, or vice versa. If someone finds a way of porting it to another platform, the licence cannot prevent this happening.

## Trick or treat? Microsoft and the Halloween Documents

What was Microsoft's response to the emergence of open-source software? Thanks to a stream of internal memos that began leaking from the company in October 1998, we have a clear idea. The documents show that the software giant had been trying to understand the concept and popularity of open source for a while and was working on ways to check its rise.

Surprisingly, Microsoft has authenticated the documents, so their content makes for interesting reading. These documents were originally leaked to open-source evangelist Eric S Raymond. He subsequently posted them, along with his comments, on the web. You can see them for yourself at [www.catb.org/~esr/halloween](http://www.catb.org/~esr/halloween).

Known as the Halloween Documents, they show that Microsoft was monitoring the spread and acceptance of open-source software with growing concern. Open source, it said, "poses a direct, short-term revenue and platform threat to Microsoft. Additionally, the intrinsic parallelism and free idea exchange in [open-source software] has benefits that are not replicable with our current licensing model."

The documents show that the size of the problem faced by proprietary software was also apparent. "The ability of the [open source] process to collect and harness the collective IQ of thousands of individuals across the internet is simply amazing," the authors wrote.

One term for a collective IQ is a 'noosphere', and Microsoft had come up with several potential strategies for exploiting such groups of individuals. One idea was to

release some of the Microsoft source code to trusted partners for study, but there were potential problems with this. The author wrote: "The challenge is to find some part of MS's code base with a big enough Noosphere to generate interest."

Another idea put forward was to exploit developers signed up to Microsoft's Developer Network by giving them a showcase to parade their Visual Basic prowess. In doing so, though, there was an assumption about why many choose to develop open source in the first place. One of the memos said: "I'll contend that many VB developers would get extreme ego gratification out of having their name/code downloadable from Microsoft.com."

One last suggestion was far more ominous. Microsoft has a well-documented history of

buying smaller, competing companies. The documents suggested that Microsoft could "monitor [open-source] news groups. Learn new ideas and hire the best/brightest individuals." If hired, talented open-source developers would inevitably have to stop working on their own projects.

Microsoft's Get the Facts campaign, which was launched after the Halloween Documents were leaked, has subsequently claimed that open-source solutions have a higher cost of ownership than proprietary software and, being community efforts, are less worthy than their proprietary rivals. Despite this, open source has increasingly trespassed on to traditionally Microsoft turf. Perhaps the only strategy is if you can't beat them, join them.



◀ Read all about it: the so-called Halloween documents show that Microsoft takes the threat posed by open source very seriously indeed

organisation) can use the code and legally maintain his own, independent code-base consisting of the original code, any modifications and any other open-source software he likes.

The most open form of development is still GNU-style. Stallman's original methodology acknowledged that many hands make light work and called whoever was interested to help push the GNU project forward. People could take the source code and do with it as they wished, while agreeing to let the world see all modifications for possible inclusion into their own work. The original code base might even include their modifications.

BSD and Apache-style development projects are perfect for dedicated teams that have good funding, time, skills and a central management structure to steer it towards well-defined goals. Those who have an idea that still needs a lot of work to develop it further, but lack the resources to do so, could struggle and fail if they insist on keeping control over the development process. In this case, a GNU-style development effort makes more sense, as it did for the Linux kernel.

GNU and Apache-style development efforts are also a good way of gathering innovative ideas from other projects that are derived from your own work. With the release of SUSE Linux 10, for example, Novell began to incorporate innovations made by others. For example, the official SUSE Linux distribution has incorporated changes made by the SUSE Performance

Enhanced Release (SUPER) project. This has halved the time it takes to boot up and pre-loads popular applications such as Firefox and OpenOffice.org. Novell has decided to incorporate these improvements into its own code base, with more changes planned.

While big software developers can clearly benefit from their involvement with the open-source movement, there are also ways that individuals can get in on the action.

### DISTRIBUTION NOW

Today there are two main ways to make money from open source. These are open to anyone and not just the owners of the source code. First, you can develop your own software and harness the skills of others to develop it further, thereby emulating the success of the Linux kernel. You can generate revenue by charging for the supply of official distribution media, documentation, training and technical support, and even tailored consultancy, while letting those with appropriate skills download it and build it themselves for free.

If your coding skills aren't up to much, there is another way. The underlying idea that open-source software is available to anyone who wants it has an interesting commercial consequence. In short, open source allows you to cherry-pick other people's ideas and package them into your own offering. As long as you redistribute these according to their original licences, you're doing nothing wrong. This seems

unfair to the original developers but, at its most basic, it is exactly what a Linux distribution is.

A good example of a company originally set up to take advantage of open source in this way is Red Hat Inc, founded by Marc Ewing. The combination of GNU software and the Linux kernel formed a free operating system, and anyone with enough programming ability could download the entire source tree for both and build them from scratch.

Ewing produced one of the first distributions of the compiled GNU/Linux operating system, along with all the source code as stipulated in the GPL. He also included automated scripts that took away much of the pain of installation. People paid a small fee for the convenience of running the installation procedure. Red Hat also offered technical support, training and documentation on a commercial basis. Five years after it was founded in 1993, Red Hat had revenues in excess of \$10 million (around £6 million). By 2005, this had risen to just under \$200 million (around £115 million).

Several other successful packages use this idea of bundling, but to a lesser extent. LAMP, for instance, consists of Linux, Apache, MySQL and PHP. Taken together, they constitute everything you need to build a highly functional, secure web server from scratch. People are free to create this bundle then give it away, while charging for secondary services such as website and e-commerce development.





# CHAPTER AND VERSE

You could also use your knowledge to turn your hand to writing a book on how to use certain pieces of open-source software. The extensive list of Linux and other open source-related titles shows that while the software is free to use, there's a large market for the knowledge necessary to make the best use of it. By including Linux Expert in the magazine, even *Computer Shopper* is benefiting from open source.

Even if you're an accomplished programmer, using and expanding on other people's open source work makes a lot of sense. According to consultancy giant Accenture, open-source software can cut development effort and project costs by up to 50 per cent.

To help you find prewritten open-source code for your project, there are a number of large repositories on the web. Among the largest are Freshmeat and SourceForge. Freshmeat ([www.freshmeat.net](http://www.freshmeat.net)) has over 40,000 projects offering nearly 200,000 code releases. It contains not only Linux software but software for other platforms including Windows and handheld devices. SourceForge ([www.sourceforge.net](http://www.sourceforge.net)) is larger, with over 100,000 registered projects. Lurking somewhere in these places are all the components you need to bring your ideas to life.

Both Freshmeat and SourceForge allow you to browse their repositories by subject. You can drill down into areas of interest to find everything from code snippets to entire projects that fulfil your needs. In addition, sites such as these provide news on new releases, new projects and general news from the world of open source. Many provide in-depth features and tutorials submitted by their users. You're free to take and use anything you like, provided you stick religiously to the licence protecting the downloaded code.

## NEW DEVELOPMENTS

The open-source movement is far from some sort of crazy anti-capitalist idea. It has handed freedom to the software it produces. This is the opposite of traditional development, which keeps source code secret and allows only a few to work on and improve it. Under open source, you give free access to software, let it develop according to the needs of a community of users and developers and sell your knowledge to those that need it.

Because of this, open-source software has found its way into businesses, local councils and even governments. It has enabled developing



◀ There are thousands of open-source projects at sites such as Freshmeat.net, so the software you need might already be out there

nations, and the charities struggling to help them, to sidestep ruinous licensing charges. Even the internet runs mostly on open-source software such as the BIND name server, Apache web server and Sendmail email server.

To increasing numbers of home and business users, software such as GIMP or OpenOffice are viable alternatives to commercial offerings. To the likes of Stallman, open-source software is free to develop and run wherever there's a need for it. Owning software isn't that important any more. No wonder Microsoft is worried.

## SAFE TO USE?

Microsoft isn't going to go away any time soon, however, and there will always be people who feel safer using its products rather than the open-source alternatives. A business might consider dumping commercial software completely, but there are always concerns about whether or not it is safe to rely on software written simply for the love of it by an ad hoc collection of developers and testers. What, for instance, happens if it suddenly stops working?

Increasingly, the first line of support is the internal IT department rather than the developer. The idea that software is free, while the knowledge to use and configure it is a commodity, has helped IT staff gain unprecedented access to software, either learning on the job or in their spare time. Add to this the growing number of training materials about open-source software, and a competent IT department has everything it needs to run Linux and its main applications.

Not only that, but such skills are in increasing demand. A quick search for Linux at

[www.monster.com](http://www.monster.com) reveals a range of job offerings, from Apache intranet developer to building and maintaining high-performance Linux clusters.

In many cases IT departments, either under pressure to work within budgets or simply because what's on offer is quick and free, are likely to be using open-source software already, using it to implement critical business functions and to monitor the state of the network. There are, however, less obvious pieces of open-source software buried right at the heart of many organisations. Sometimes those high up in the organisation might not even know about the open-source software powering the company.

For example, even small companies now run their own internal DNS servers to speed up DNS lookups, rather than using the often overloaded servers supplied by their ISPs. These internal DNS servers use the free BIND software, which is often installed on an old machine located somewhere in a corner of the server room.

Similarly, it's likely that the company's firewall is Linux-based, too. Space rented from a third party to host a corporate website, even if it exists on a Microsoft Windows machine, is also likely to use Apache to serve content.

The website-hosting company itself might even be completely Linux based, thereby making licensing savings it can pass on to the customer that it could never hope to do as a Microsoft-only concern. In the dedicated data centre, running Linux file servers is now commonplace. With Samba, for instance, desktop users can store their data on Linux servers as seamlessly as if it were a Windows server and they would not know the difference.

Though there might be an argument that niche open-source applications could be problematic from a support point of view, that's not the case with well-known packages. Open source is big business. It's now the large software producers such as Novell or IBM, rather than the informal groups of enthusiasts of yesteryear, that compile and release Linux distributions. These suppliers have the influence, commitment and development capability needed to ensure that open-source software will continue its advance into the mainstream. ☐

▶ By browsing an online open-source code repository, you could find all the components you need to turn your idea into reality without becoming an accomplished and prolific programmer first



## Next Month

### VIRTUAL MACHINES

How to get one PC to emulate a different PC or set of PCs